AMENDMENTS TO THE SPECIFICATION

Please amend the specification, as follows:

Replace paragraph [0005] with the following amended paragraph [0005]:

A Montgomery modular multiplication algorithm, known as the most effective modular multiplication algorithm, can be expressed in pseudo code, as in Algorithm 1 below:

[Algorithm 1]

Stimulus:

$A = (a_{n-1}\ a_{n-2} \ldots a_1\ a_0)_2$, and $A < M$

$B = (b_{n-1}\ b_{n-2} \ldots b_1\ b_0)_2$, and $B < M$

$M = (m_{n-1}\ m_{n-2} \ldots m_1\ m_0)_2$, and M is odd[[.]]

Response:

$S = \sout{(S_n\ S_{n-1}\ S_{n-2} \ldots S_1\ S_0)}\ \underline{(s_n\ s_{n-1}\ s_{n-2} \ldots s_1\ s_0)_2} \equiv ABR^{-1} \pmod{M}$

Method:

S: = 0

For i: = 0 to n-1 do

[[qi]] $\underline{q_i}$: = s_0 XOR (b_i AND a_0)

S: = $(S + b_iA + q_iM)/2$

endfor

Replace paragraph [0018] with the following amended paragraph [0018]:

The q_i calculation logic circuit solves a Boolean logic equation "s_0 XOR c_0 XOR (b_i AND a_0)", where s_0 is the least significant bit (LSB) of a sum S, c_0 is the LSB of a carry C, b_i is the bit value of the number B, and a_0 is the LSB of the number A, to obtain a bit value q_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1).

Replace paragraph [0030] with the following amended paragraph [0030]:

Another exemplary embodiment of the present invention[[,]] provides a method of performing a Montgomery modular multiplication in a Montgomery modular multiplier, which includes registers for storing bit values a_i, b_i, m_i, c_i, and s_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) of a word A, a word B, a modulus M, a carry C, and a sum S, respectively, and calculates a value congruent to "ABR^{-1}" (mod M), where A and B are input n-bit numbers, R^{-1} is an inverse number of R modular-multiplied for "mod M", and M is a modulus. In the method, the number A, the number B, and the modulus M are received. The number A is multiplied by a bit value b_i to obtain each bit of b_iA. A Boolean logic equation "s_0 XOR c_0 XOR (b_i AND a_0)", where s_0 is the least significant bit (LSB) of a sum S, c_0 is the LSB of a carry C, b_i is the bit value of the number B, and a_0 is the LSB of the number A, is obtained to obtain a bit value q_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1). The number M is multiplied by the bit value q_i to obtain each bit of q_iM. Then, 'n' additions are performed on the carry C, the sum S, the b_iA, and the q_iM to obtain interim values for each bit of the sum S and the carry C in a carry save adder structure, in response to a carry propagation adder signal.

The interim values are summed to obtain the final results of the sum S and carry C in a carry propagation adder structure, in response to the carry propagation adder signal.

Replace paragraph [0045] with the following amended paragraph [0045]:

The A-register 110 stores the bit value a_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) of the number A, which is smaller than the modulus M. The number A denotes a word representing an input n-bit number, and a_i is the value of each of the bits a_0 to a_{n-1} that constitute the number A.

Replace paragraph [0046] with the following amended paragraph [0046]:

The B-register 120 stores the bit value b_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) of the number B, which is smaller than the modulus M. The number B denotes a word representing an input n-bit number, and b_i is the value of each of the bits b_0 to b_{n-1} that constitute the number B.

Replace paragraph [0047] with the following amended paragraph [0047]:

The M-register 130 stores the bit value m_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) of the modulus M, which is an odd number. The modulus M denotes a word representing an input n-bit number, and m_i is the value of each of the bits m_0 to m_{n-1} that constitute the modulus M.

Replace paragraph [0048] with the following amended paragraph [0048]:

The b_iA calculation logic circuit 140 calculates each bit of b_iA by multiplying the number A by the bit value b_i. Consequently, the values of the 'n' bits b_ia_0 to b_ia_{n-1} are output. At this time, since 'i' varies from 0 to [[n*1]] n-1 in the "for" loop included in Algorithm 1, the value b_i is obtained from the position of the least significant bit (LSB) of the B-register 120, which is right shifted by one bit every time an algorithm in the "for" loop is performed, as shown in FIG. 1.

Replace paragraph [0049] with the following amended paragraph [0049]:

The q_i calculation logic circuit 150 calculates the value q_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) used in the "for" loop of Algorithm 1 by solving the Boolean logic equation "s_0 XOR c_0 XOR (b_i AND a_0)". Here, s_0 is the LSB of a sum S, c_0 is the LSB of a carry C, b_i is a bit value of the number B, and a_0 is the LSB of the number A. At this time, since 'i' varies from 0 to [[n*1]] n-1 in the "for" loop included in Algorithm 1, a value b_i is obtained from the position of the LSB of the B-register 120, which is right shifted by one bit every time an algorithm in the "for" loop is performed, as shown in FIG. 1.

Replace paragraph [0050] with the following amended paragraph [0050]:

The q_iM calculation logic circuit 160 calculates each bit of q_iM by multiplying the modulus M by the bit value q_i. Consequently, the values of the 'n' bits q_im_0 to q_im_{n-1} are output. At this time, since 'i' varies from 0 to [[n*1]] n-1 in the "for" loop included in Algorithm 1, 'i' increases by one every time an algorithm in the "for" loop is performed, as shown in FIG. 1. Consequently, the values of the 'n' bits q_0 to q_{n-1} are output.

Replace paragraph [0052] with the following amended paragraph [0052]:

The S-register 180 updates and stores the bit value s_i of the sum S (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1). In other words, sum S denotes a word representing an n-bit number that is output as a sum, and s_i denotes the value of each of the bits s_0 to s_{n-1} that constitute the word S. The word S is updated every time an addition is performed in the carry save adder or carry propagation adder included in the 4-2 compressor 170.

Replace paragraph [0053] with the following amended paragraph [0053]:

The C-register 190 updates and stores the bit value c_i of the carry C (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1). In other words, carry C denotes a word representing an n-bit number that is output as a carry, and c_i denotes the value of each of the bits c_0 to c_{n-1} that constitute the word C. The word C is updated every time an addition is performed in the carry save adder or carry propagation adder included in the 4-2 compressor 170.

Replace paragraph [0061] with the following amended paragraph [0061]:

The Montgomery modular multiplier according to an embodiment of the present invention includes registers, which store bit values a_i, b_i, m_i, c_i, and s_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) of a word A, a word B, a modulus M, a carry C, and a sum S, respectively, and calculates a value congruent to "ABR^{-1}" (mod M). Here, A and B are input n-bit numbers, and R^{-1} is an inverse number of R modular-multiplied for "mod M [[A]]".

Replace paragraph [0063] with the following amended paragraph [0063]:

Thereafter, in step S315 to S319, the q_i calculation logic circuit 150 of the Montgomery modular multiplier obtains a value q_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) used in the "for" loop of Algorithm 1, by solving the Boolean logic equation "s_0 XOR c_0 XOR (b_i AND a_0)". Here, s_0 is the LSB of a sum S, c_0 is the LSB of a carry C, b_i is a bit value of the number B, and a_0 is the LSB of the number A. Also, in steps S315 to S319, the b_iA calculation logic circuit 140 multiplies the number A by the bit value b_i to obtain each bit of b_iA, and the q_iM calculation logic circuit 160 calculates each bit of q_iM by multiplying the modulus M by the bit value q_i. Also, in steps S315 to S319, the 4-2 compressor 170 performs 'n' additions on the carry C, the sum S, the b_iA, and the q_iM to obtain interim values for each bit of the sum S and the carry C, in a carry save adder structure, which is formed when the carry propagation adder signal ONCPA is in an inactive state, that is, is in a first logic state ("0").

Replace paragraph [0069] with the following amended paragraph [0069]:

As described above, the Montgomery modular multiplier according to an embodiment of the present invention includes registers, which store bit values a_i, b_i, m_i, c_i, and s_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) of a word A, a word B, a modulus M, a carry C, and a sum S, respectively, and calculates a value congruent to "ABR^{-1}" (mod M). Here, A and B are input n-bit numbers, and R^{-1} is an inverse number of R modular-multiplied for "mod A". The b_iA calculation logic circuit 140 calculates each bit of b_iA by multiplying the number A by the bit value b_i. At this time, the q_i calculation logic circuit 150 calculates a value q_i (where 'i' denotes an integer in the range of 0 to [[n*1]] n-1) by solving a Boolean logic equation "s_0 XOR c_0 XOR (b_i AND a_0)". Here, s_0 is the LSB of a sum S, c_0 is the LSB of a carry C, b_i is a bit value of the number B, and a_0 is the LSB of the number A. The q_iM calculation logic circuit 160 calculates each bit of q_iM by multiplying the modulus M by the bit value q_i. In response to the carry propagation adder signal ONCPA, the 4-2 compressor 170 performs 'n' additions on the carry C, the sum S, the b_iA, and the q_iM to obtain interim calculated values for each bit of the sum S and the carry C, in a carry save adder structure. Then, the 4-2 compressor 170 sums the interim calculated values to obtain the final results of the sum S and carry C in a carry propagation adder structure. The final results of the sum S and carry C are output to the S- and C-registers 180 and 190, respectively.